P01-030705

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年　９月１７日

出願番号
Application Number: 特願２００２－２７０３２４

[ST.10/C]: [ＪＰ２００２－２７０３２４]

出願人
Applicant(s): パイオニア株式会社

２００３年　６月２６日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３０５０５３２

【書類名】　特許願

【整理番号】　57P0127

【提出日】　平成14年 9月17日

【あて先】　特許庁長官　殿

【国際特許分類】　H04L 1/00

H04L 13/00

H03M 13/00

【発明者】

【住所又は居所】　東京都大田区大森西４丁目１５番５号　パイオニア株式会社　大森工場内

【氏名】　勝屋　宏一

【特許出願人】

【識別番号】　000005016

【氏名又は名称】　パイオニア株式会社

【代理人】

【識別番号】　100083839

【弁理士】

【氏名又は名称】　石川　泰男

【電話番号】　03-5443-8461

【手数料の表示】

【予納台帳番号】　007191

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9102133

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　フレーム構造のノイズ除去装置、フレーム構造のノイズ除去方法およびフレーム構造のノイズ除去プログラム

【特許請求の範囲】

【請求項１】　符号化された音声データにエラーが発生したかどうかを検出するエラー検出手段と、

前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコード手段を備え、

前記デコード手段は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする、ノイズ除去装置。

【請求項２】　請求項１に記載のノイズ除去装置において、

前記エラー検出手段は、特定の放送仕様に使用される記述子と一致するか否かを判定することを特徴とする、ノイズ除去装置。

【請求項３】　請求項１または２に記載のノイズ除去装置において、

前記エラー検出手段は、前記符号化された音声データに含まれるデータ長記述子を利用し、前記符号化された音声データにエラーが発生したか否かを判定することを特徴とする、ノイズ除去装置。

【請求項４】　請求項１から３のいずれかに記載のノイズ除去装置において、

前記デコード手段は、前記エラー検出時にデコード結果が０となる前記符号化された音声データをデコードすることを特徴とする、ノイズ除去装置。

【請求項５】　符号化された音声データにエラーが発生したかどうかを検出するエラー検出ステップと、

前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコードステップを備え、

前記デコードステップは、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする、ノイズ除去方法。

【請求項６】　請求項５に記載のノイズ除去方法において、

前記エラー検出ステップは、特定の放送仕様に使用される記述子と一致するか否かを判定することを特徴とする、ノイズ除去方法。

【請求項７】　請求項５または６に記載のノイズ除去方法において、

前記エラー検出ステップは、前記符号化された音声データに含まれるデータ長記述子を利用し、前記符号化された音声データにエラーが発生したか否かを判定するステップであることを特徴とする、ノイズ除去方法。

【請求項８】　請求項５から７のいずれかに記載のノイズ除去方法において、

前記デコードステップは、前記エラー検出時にデコード結果が０となる前記符号化された音声データをデコードするステップであることを特徴とする、ノイズ除去方法。

【請求項９】　コンピュータに、符号化された音声データにエラーが発生したかどうかを検出するエラー検出機能と、

前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコード機能を備えさせ、

前記デコード機能は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを実現させるためのプログラム。

【請求項１０】　請求項９に記載のプログラムにおいて、

前記エラー検出機能は、特定の放送仕様に使用される記述子と一致するか否かを判定することを特徴とする、プログラム。

【請求項１１】　請求項９または１０に記載のプログラムにおいて、

前記エラー検出機能は、前記符号化された音声データに含まれるデータ長記述子を利用し、前記符号化された音声データにエラーが発生したか否かを判定する機能であることを特徴とする、プログラム。

【請求項１２】　請求項９から１１のいずれかに記載のプログラムにおいて、

前記デコード機能は、前記エラー検出時にデコード結果が０となる前記符号化された音声データをデコードすることを特徴とする、プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、デジタルオーディオデータフレーム、圧縮音声データフレームその他のフレーム構造のノイズ除去に関する。

【０００２】

【従来の技術】

【特許文献１】特開平８－２８６６９８

【特許文献２】特開２０００－５９２３１

【特許文献３】特表２００１－５０１０６３

【特許文献４】特開２００２－７３０９１

従来の圧縮音声再生方法においては、圧縮音声ストリーム中のフレームエラーを検出する方法としては、ＣＲＣ（Ｃｙｃｌｉｃ　Ｒｅｄｕｎｄａｎｃｙ　Ｃｈｅｃｋ）によるエラーチェックが一般的である。圧縮音声再生方法の一つであるＡＡＣ（Ａｄｖａｎｃｅｄ　Ａｕｄｉｏ　Ｃｏｄｉｎｇ）の場合、ＣＲＣでエラー検出対象としている範囲は、圧縮音声ストリームの全体ではなく、一部分である。圧縮音声ストリームはフレームの集合からなるが、ＩＳＤＢ－ＴＳＢ（Ｉｎｔｅｇｒａｔｅｄ　Ｓｅｒｖｉｃｅｓ　Ｄｉｇｉｔａｌ　Ｂｒｏａｄｃａｓｔｉｎｇ－Ｔｅｒｒｅｓｔｒｉａｌ　Ｓｏｕｎｄ　Ｂｒｏａｄｃａｓｔｉｎｇ）：（国内ディジタル地上波オーディオ放送）ではＡＡＣのなかのＡＤＴＳ（Ａｕｄｉｏ　Ｄａｔａ　Ｔｒａｎｓｐｏｒｔ　Ｓｔｒｅａｍ）フレームを使用している。図１はＡＤＴＳフレームの構造を示す。ＡＤＴＳフレームはＡＤＴＳヘッダー、ＣＲＣ、Ｒａｗ＿ｄａｔａ＿ｂｌｏｃｋの３つの部分からなる。ＡＤＴＳヘッダーにはフレームの各種情報が書き込まれている。ＣＲＣにはＡＤＴＳフレームの一部分についてエラーチェックをした結果が書き込まれている。Ｒａｗ＿ｄａｔａ＿ｂｌｏｃｋには圧縮音声データおよび圧縮音声データの種類を示す情報が書き込まれている。図２はＲａｗ＿ｄａｔａ＿ｂｌｏｃｋの構造を示す。Ｒａｗ＿ｄａｔａ＿ｂｌｏｃｋは、圧縮音声データの種類を示すＩＤ、圧縮音声データであるＳｙｎｔａｃｔｉｃ　Ｅｌｅｍｅｎｔ、その他データであるｂｙｔｅ　ａｌｉｇｎｍｅｎｔから構成されている。ＩＳＤＢ－ＴＳＢのチャンネルの構成やプ

ロファイルにより、ＩＤの種類や数が異なる。図３にＩＤの種類を示す。

【０００３】

図３には左欄から、Ｓｙｎｔａｃｔｉｃ　Ｅｌｅｍｅｎｔの名称、ＩＤの名称、ＩＤのコード、略語の順に８種類が記載されている。上から３番目のＩＤ＿ＣＣＥはＩＳＤＢ－ＴＳＢでは使用されない。

【０００４】

図４に従来のＡＡＣデコーダのエラー検出方法のフローチャートを示す。ステップＳ１では音声圧縮ストリーム中の１フレームが入力される。ステップＳ２では、ステップＳ１において入力された１フレーム中のヘッダー部分を取得する。ヘッダー部分には、ＩＤ、レイヤ、保護ビット、プロファイル、およびサンプリング周波数等の情報が格納されている。ステップＳ３では、ステップＳ２で取得したヘッダー情報がＡＡＣの仕様に一致しているかいないかを判断する。ヘッダー情報がＡＡＣの仕様に一致している場合にはステップＳ４へ進み、一致していない場合にはステップＳ８に進む。

【０００５】

ステップＳ４では、ＡＤＴＳフレームのうちＲａｗ＿Ｄａｔａ＿Ｂｌｏｃｋ部分を取得する。ここには図２で説明した各種ＩＤとＳｙｎｔａｃｔｉｃ　Ｅｌｅｍｅｎｔが格納されている。図３に表示したＩＤと同一のものだけが含まれる場合にはステップＳ６に進み、図３に表示したＩＤと異なるものが含まれる場合には、ステップＳ８に進む。すなわち、各ＩＤの種類を示すコードが０Ｘ０～０Ｘ７に該当する場合には、ステップＳ６に進み、コードが０Ｘ０～０Ｘ７に該当しない場合には、ステップＳ８に進む。

【０００６】

ステップＳ６ではＣＲＣチェックを行う。ＡＡＣのＡＤＴＳフレームにおいてＣＲＣチェックの対象となるのは、ＡＤＴＳヘッダーの全てと、図２のＩＤのうちＳＣＥ、ＣＰＥ、ＣＣＥ、ＬＦＥの最初の１９２ビットと、ＣＰＥ内にある２番目のｃｈａｎｎｅｌ＿ｐａｉｒ＿ｅｌｅｍｅｎｔの最初１２８ビットと、ＰＣＥ、ＤＳＥの全てのデータである。これらのＣＲＣのチェック対象部分を生成多項式にいれて計算し、ＡＤＴＳヘッダー直後のＣＲＣの値と比較して、値が一致

すればフレーム内にエラーがなかったとしてステップＳ７に進み、値が一致しなければエラーがフレーム内にあるものとしてステップＳ８に進む。従って、各ＩＤ、ｆｉｌｌ＿ｅｌｅｍｅｎｔ、およびＳＣＥ、ＣＰＥ、ＣＣＥ、ＬＦＥの１９３ビット以降、ＣＰＥ内にある２番目のｃｈａｎｎｅｌ＿ｐａｉｒ＿ｅｌｅｍｅｎｔの最初から１２９ビット以降の部分はエラーチェックの対象とはなっていない。

【０００７】

ステップＳ７ではフレーム内にエラーが発生していないＡＡＣフレームについてはデコード処理を行う。

【０００８】

ステップＳ８では、フレーム内でエラーが発生したと判断された場合の処理を行う。フレーム内でエラーが発生したと判断された場合には、デコード処理を行わずにデコード結果のデータを全て“０”で置き換える。

【０００９】

この場合に、エラーが発生したフレームは、“０”として出力し、ソフトミュート（フェードイン／フェードアウト）を施すことにより、エラー発生時における聞きにくさを解消しようとする（第１の従来技術）。その他にも、エラー発生時における聞きにくさを解消するために前のデータを繰り返すなどの方法（第２の従来技術）が採用されている。

【００１０】

図５および図６を用いてこれらの場合について説明する。図５は第１の従来技術を示す概略図であり、図６は第２の従来技術を示す概略図である。図５の上段は、入力データ、中段は、入力データのデコーダ、下段は出力データを示す。入力はフレーム毎にＣＲＣが付加されており、図１ではフレーム０、フレーム１、フレーム２の３つが図示されている。フレーム１のＣＲＣにおいてエラーが発見された場合、フレーム１をデコードした結果のフレーム１Ａのデータは、すべて“０”とされる。その結果、フレーム０Ａとフレーム２Ａ間でデータが途切れるので、フレーム０Ａのフレーム１Ａに近い部分の出力を徐々に小さくしていき（フェードアウト）、フレーム２Ａのフレーム１Ａに近い部分を小さい出力から徐

々に大きくしていく（フェードイン）。このようにして、エラー発生時における聞きにくさを解消する方法がとられる。

【００１１】

図６の上段は、入力データ、中段は、入力データのデコーダ、下段は出力データを示す。入力はフレーム毎にＣＲＣが付加されており、図６ではフレーム３、フレーム４、フレーム５の３つが図示されている。フレーム４のＣＲＣにおいてエラーが発見された場合、フレーム４をデコードした結果のフレーム４Ａのデータは、エラーが発生していない直前のフレーム３Ａのデータにすべて置き換えられる。このようにして、エラー発生時における聞きにくさを解消する方法がとられる。

【００１２】

【発明が解決しようとする課題】

図４に示す従来技術のエラー検出方法においては、ＣＲＣでエラー検出ができるデータはフレームの一部のみであるので、エラー検出能力が低いという問題点を有していた。また、図５に示すソフトミュート（フェードイン／フェードアウト）によるエラー処理を行っても、出力音が有音（正常フレーム）から無音（エラー発生フレーム）に変化し、更に有音（正常フレーム）に変化することになるので、聴きずらいと感じる場合もあった。また、図６において説明した、エラー検出前のフレームのデータを使用する場合にも、エラー検出フレームの前のフレーム音が間延びしたような感覚や、エラー検出フレームの直後のフレーム音が音飛びしたような感覚を生じる場合があった。

【００１３】

そこで、本発明では、ＣＲＣによるエラー検出よりもさらにエラー検出能力を高め、かつ、エラー検出時における出力音をより聞きやすい音に改善することを課題の一例としている。

【００１４】

【課題を解決するための手段】

上記の課題を解決するために、請求項１に記載の発明は、符号化された音声データにエラーが発生したかどうかを検出するエラー検出手段と、前記符号化され

た音声データをデコードする際に窓関数を使用し、その結果を加算するデコード手段を備え、前記デコード手段は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする。

【００１５】

【発明の実施の形態】

－第１の実施形態－

以下、ノイズ除去装置による第１の実施形態について説明する。図７は、本実施形態のエラー検出方法をフローチャートで示したものである。

【００１６】

本実施形態ではＩＳＤＢ－ＴＳＢ（国内デジタル地上波オーディオ放送）において使用されているＡＡＣにおけるＡＤＴＳフレームの処理について説明を行う。

【００１７】

ステップＳ９ではデータストリーム中の１フレームが入力される。ステップＳ１０では、ステップＳ９において入力された１フレーム中のヘッダー部分を取得する。ヘッダー部分には、ＩＤ、レイヤ、保護ビット、ビットレート、およびサンプリング周波数等の情報が格納されている。ステップＳ１１では、ステップＳ１０で取得したヘッダー情報がＩＳＤＢ－ＴＳＢの仕様を決めているＡＲＩＢ（Ａｓｓｏｃｉａｔｉｏｎ ｏｆ Ｒａｄｉｏ Ｉｎｄｕｓｔｒｉｅｓ ａｎｄ Ｂｕｓｉｎｅｓｓｅｓ）の仕様に一致しているかいないかを判断する。ヘッダー情報がＡＲＩＢの仕様に一致している場合にはステップＳ１２へ進み、一致していない場合にはステップＳ１７に進む。たとえば、サンプリング周波数について、ＡＡＣの規格では１２種類が定義されているが、ＩＳＤＢ－ＴＳＢのＡＲＩＢではそのうち４８ｋＨｚ、３２ｋＨｚ、２４ｋＨｚの３種類のみしか使用していないので、サンプリング周波数が４８ｋＨｚ、３２ｋＨｚ、２４ｋＨｚの場合にはステップＳ１２へ進み、サンプリング周波数が４８ｋＨｚ、３２ｋＨｚ、２４ｋＨｚでない場合にはステップＳ１７へ進むことになる。

【００１８】

ステップＳ１２では、ＡＤＴＳフレームのうちＲａｗ＿Ｄａｔａ＿Ｂｌｏｃｋ

部分を取得する。ここには図３で説明した各種ＩＤとＳｙｎｔａｃｔｉｃ　Ｅｌｅｍｅｎｔが格納されている。ステップＳ１３では各種ＩＤのうちＩＳＤＢ－ＴＳＢで使用していないＩＤをＲａｗ＿Ｄａｔａ＿Ｂｌｏｃｋ部分にもつ場合にはＳ９に進み、ＩＳＤＢ－ＴＳＢで使用しているＩＤをＲａｗ＿Ｄａｔａ＿Ｂｌｏｃｋ部分にもつ場合にはステップＳ１４に進む。

【００１９】

ステップＳ１４では、ＣＲＣ（Ｃｙｃｌｉｃ　Ｒｅｄｕｎｄａｎｃｙ　Ｃｈｅｃｋ）によってエラーが発生しているかどうかをチェックする。このＣＲＣチェックは従来から実施されていたものである。ＣＲＣによってエラーが検出された場合にはステップＳ１７に進み、ＣＲＣによってエラーが検出されなかった場合にはステップＳ１５に進む。

【００２０】

ステップＳ１５では、ＡＤＴＳヘッダーに書き込まれているフレーム長さ情報に基づいて、処理したフレーム全体の長さがフレーム長さ情報どおりか否かを判断している。すなわち、ヘッダーの先頭から処理したビット数をカウントし、図２に示すｂｙｔｅ　ａｌｉｇｎｍｅｎｔの最後のビットまで計算する。そのビット数がＡＤＴＳヘッダーに書き込まれているフレーム長さ情報と一致するか否かを判断する。一致しない場合は、エラーがあるものとみなして、ステップＳ１７に進み、一致する場合は、エラーがないものとみなして、ステップＳ１６に進む。ステップＳ１６に進む場合には、取得しているフレームの内容をメモリに書き込む。

【００２１】

ステップＳ１６では、取得したフレームをデコード処理する。この動作について図８を参照して説明する。図８の上半分はエンコード処理を表わし、下半分はここで説明するデコード処理を表わす。周波数サンプル列９内のフレーム２１、２２、２３がステップＳ９で取得する各フレームに対応する。デコード処理の最初は、各フレームに逆変形離散コサイン変換ＩＭＤＣＴ（Ｉｎｖｅｒｓｅ　Ｍｏｄｉｆｉｅｄ　Ｄｉｓｃｒｅｔｅ　Ｃｏｓｉｎｅ　Ｔｒａｎｓｆｏｒｍ）１０処理を行う。ＩＭＤＣＴ１０の変換式は以下のように表わされる。

【００２２】

【数１】

$$Xi,n=2/N\Sigma\ (\text{from0 to } N/2-1)\ \text{spec}[i][k]\cos(2\pi/N(n+n0)(k+1/2))$$

$0 \leqq n < N$、（n：sample index, i：windows index, k：spectral coefficient index, N：window length based on the window sequence value, n0=(N/2+1)/2）

次に、ＩＭＤＣＴ１０の出力ブロック２１Ａ、２２Ａ、２３Ａに窓関数１１をかける。この窓関数１１は一種のフィルタとして考えることもできる。各フレームは周波数特性をもつが、窓関数１１によりこの周波数特性が決定される。また、窓関数１１を用いることにより前後のブロックの連続性を確保する。ＡＡＣでは、窓関数１１としてサイン窓と隣接バンドとの分離度の良いガイザー窓の２種類が定義されており、本実施形態ではいずれの窓関数をも選択することが可能である。

【００２３】

窓関数１１の処理は一のＩＭＤＣＴ１０の出力ブロックと、その隣接する両ブロックのブロックサイズの半分ずつをオーバラップさせて行う。図８においてはブロック２２Ａはその隣接するブロック２１Ａとブロック２３Ａのブロックサイズの半分づつをオーバラップさせて窓関数１１の処理を行っている。その後、窓関数１１の処理を行った出力ブロックの隣接するブロックのブロックサイズの半分ずつを加算１２して、信号を再生し、時間サンプル列１３を生成する。

【００２４】

したがって、フレーム２２にエラーがあって、デコード結果が０となっても、フレーム２２のデコード結果が一部となる対応する時間サンプル列ブロック２５および２６には、ブロック２１Ａの後半部と、ブロック２３Ａの前半部がデータとして入るので、無音となることはなく、しかも前後のブロック（ブロック２１とブロック２３）からのデコード処理結果がデータとして入るので、ブロック２２と相関した連続性を確保したデータが入ることになる。

【００２５】

ステップＳ１７では、ステップＳ９からステップＳ１５までの間のエラーチェ

ックにおいて、フレーム内にエラーが発生していないと判断されたフレームが記憶される。したがって、フレーム内にエラーが発生した場合に、ステップＳ１６にてデコード処理が行われるフレームは、ステップＳ１７に記憶されている、エラーが発生していない直前のフレームとなる。フレームにエラーが発生した場合でもステップＳ１６のデコード処理について説明したように、窓関数１１の処理は前後のフレームのデータを使用しているので、前後のデータの連続性は確保されており、再生された信号の音が突然途切れることはない。

【００２６】

ＡＡＣはブロック符号化を行っていることから、デコーダで時間信号に戻したときに、圧縮歪みはブロック内に歪みとして広がる。ブロックごとに圧縮の仕方が異なれば、時間信号にもどしたときにブロック間で不連続が発生し、ブロック歪みとよばれる歪みが発生する。オーディオの場合、この非連続なブロック歪みの音は人間に不快感を与えやすい。このため、ステップＳ１７において、オーバラップして窓関数１１処理をして前後のデータの連続性を確保して滑らかにブロック間をつなぐことにより、ブロック歪みが軽減されることになる。

－第２の実施形態－

以下、ノイズ除去装置による第２の実施形態について説明する。

【００２７】

図９は、本実施形態のエラー検出方法をフローチャートで示したものである。第１の実施形態と重複する部分は同一符号を付した。

【００２８】

第２の実施形態は、第１の実施形態とステップＳ１８における処理の内容が異なる。ステップＳ１８において記憶されるフレームデータはステップＳ１６のデコード処理の結果が０となるフレームデータである。すなわち、図９中ステップＳ１１、Ｓ１３、Ｓ１４、Ｓ１５においてフレームにエラーがあると判断された場合には、当該フレームのデコード結果は０になる。しかし、ステップＳ１６における窓関数処理は一のＩＭＤＣＴの出力ブロックと、その隣接する両ブロックのブロックサイズの半分ずつをオーバラップさせた処理を行う。すなわち、図８において、ブロック２２Ａはその隣接するブロック２１Ａとブロック２３Ａのブ

ロックサイズの半分づつをオーバラップさせて窓関数処理を行う。そして、窓関数処理を行った出力ブロックの隣接するブロックのブロックサイズの半分ずつを加算して出力信号を再生する。したがって、当該フレーム出力結果が０であっても、当該フレームの前後のフレームをデコードした結果が当該フレームのデコード結果としての出力信号になる。すなわち、デコードされた信号は、従来技術のように有音（正常フレーム）から無音（エラー発生フレーム）へ、無音（エラー発生フレーム）から有音（正常フレーム）へと変化するのではなく、音量は低下するが有音状態が連続し、しかも相互に関連する信号として取り出すことができる。このように、ステップＳ１８において、デコード結果が０になるようなデータを記憶しておくことにより、オーバラップした窓関数処理を行って、前後のデータの連続性を確保して滑らかにブロック間をつなぎ、ブロック歪みが軽減されることになる。このため、人間に不快感を与えにくくなる。

－第３の実施形態－

以下、ノイズ除去装置による第３の実施形態について説明する。

【００２９】

図１０は、本実施形態のエラー検出、およびデコード処理をブロック図で示したものである。エラー検出部１４、メモリ１５～１７、セレクタ部１９、デコード処理部２０からなる。入力データとして１フレームずつ、エラー検出部１４とメモリ１６に入力される。エラー検出部では、図９におけるステップＳ１１、Ｓ１３、Ｓ１４、Ｓ１５の処理が順次行われる。もしフレーム中にエラーがあると判断した場合には、入力切替信号１８によってセレクタ部１９はメモリ１５の内容またはメモリ１７の内容をデコード処理部２０に出力する。メモリ１５にはエラーが発生していない直前のフレームのデータが記憶されており、メモリ１７にはデコード結果が０になるフレームパターンが記憶されている。エラー検出部においてフレームにエラーが発生していないと判断した場合には、入力切替信号１８によってセレクタ部１９はメモリ１６の値をデコード処理部２０に出力する。メモリ１６には現在エラー検出を行っているフレームが記憶されているので、フレーム中にエラーが発生していないと判断した場合には、現在のフレームをデコードする通常の処理が行われる。

【００３０】

すなわち、入力切替信号１８によってセレクタ部１９はメモリ１５～１７のいずれかの値をデコード処理部２０に出力する。

【００３１】

デコード処理部２０では、取得したフレームをデコード処理する。このデコード処理について図８を参照して説明する。図８の下半分はここで説明するデコード処理をあらわす。周波数サンプル列がメモリ１６で取得したフレームに対応する。デコードするフレームに逆変形離散コサイン変換ＩＭＤＣＴ処理を行う。

【００３２】

次に、ＩＭＤＣＴの出力であるブロックに窓関数をかける。この窓関数は一種のフィルタとして考えることもでき、各フレームは周波数特性をもつが、窓関数によりこの周波数特性が決定される。ＡＡＣでは、窓関数としてサイン窓と隣接バンドとの分離度の良いガイザー窓の２種類が定義されており、本実施形態ではいずれの窓関数をも選択することが可能である。

【００３３】

窓関数処理は一つのＩＭＤＣＴの出力ブロックと、その隣接する両ブロックのブロックサイズの半分ずつをオーバラップさせて窓関数処理を行う。図８においてブロックは隣接する前後のブロックのブロックサイズの半分づつをオーバラップさせて窓関数処理を行っている。窓関数処理を行った出力ブロックの隣接するブロックのブロックサイズの半分ずつを加算して、信号を再生する。

【００３４】

したがって、フレームでエラーが検出された場合、メモリ１５の内容（エラーが発生していない直前のフレームデータ）または、メモリ１７の内容（デコード処理の結果が“０”となるフレームデータ）をデコード処理するが、いずれの場合もエラーが発生したフレームをデコード処理した結果が無音となることはなく、しかも前後のフレームから出力音を再生するので、前後のデータの連続性が確保されており、エラー検出時における出力音をより聞きやすい音に改善することができる。

【００３５】

また、本出願によるノイズ除去方法はノイズ除去装置によって実施される図７および図８において明示されている。

【００３６】

さらに、上記図７および図９に示すフローチャートに対応するプログラムをフレキシブルディスクまたはハードディスク等の媒体に記録しておき、或いはインターネット等のネットワークを用いて配信しておき、これをマイクロコンピュータ等により読み出して実行することにより、当該マイクロコンピュータ等をシステムコントローラとして動作させることも可能である。

【００３７】

本出願にかかわる実施形態は、ＭＤＣＴを使用しているＭＰ３、ＡＣ－３、ＭＰＥＧ４あるいはＡＴＲＡＣなどにも適応することができる。

【図面の簡単な説明】

【図１】

ＡＤＴＳフレームの構造を示す図。

【図２】

Ｒａｗ＿ｄａｔａ＿ｂｌｏｃｋの構造を示す図。

【図３】

ＩＤの種類を示す図。

【図４】

従来のＡＡＣデコーダのエラー検出方法のフローチャート。

【図５】

第１の従来技術を示す概略図。

【図６】

第２の従来技術を示す概略図。

【図７】

第１の実施形態のエラー検出方法フローチャート。

【図８】

エンコード処理およびデコード処理の手順。

【図９】

第２の実施形態のエラー検出方法のフローチャート。

【図１０】

第３の実施形態のエラー検出およびデコード処理のブロック図。

【符号の説明】

６、１３：時間サンプル列

９：周波数サンプル列

１０：ＩＭＤＣＴ

１１：窓関数

１２：加算

１４：エラー検出部

１５、１６、１７：メモリ

１８：入力切替信号

１９：セレクタ部

２０：デコード処理部

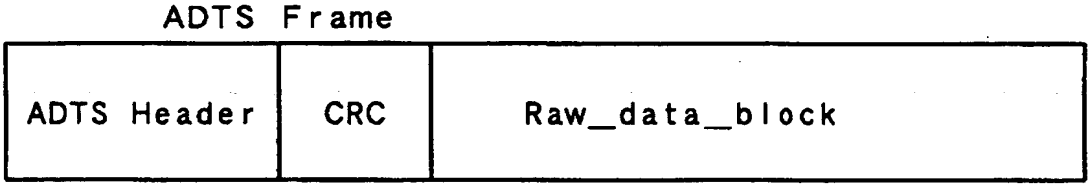
２１、２２、２３：フレーム

２１Ａ、２２Ａ、２３Ａ：ブロック

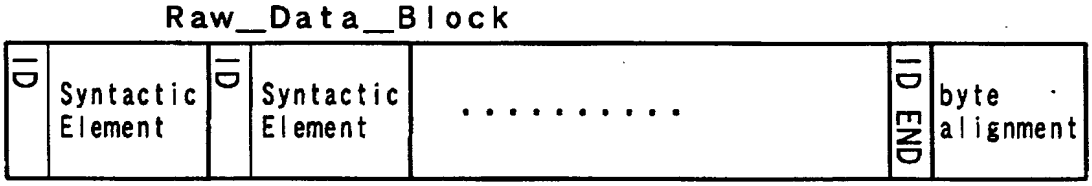
２４：システムコントローラ

【書類名】　　図面

【図１】



【図２】

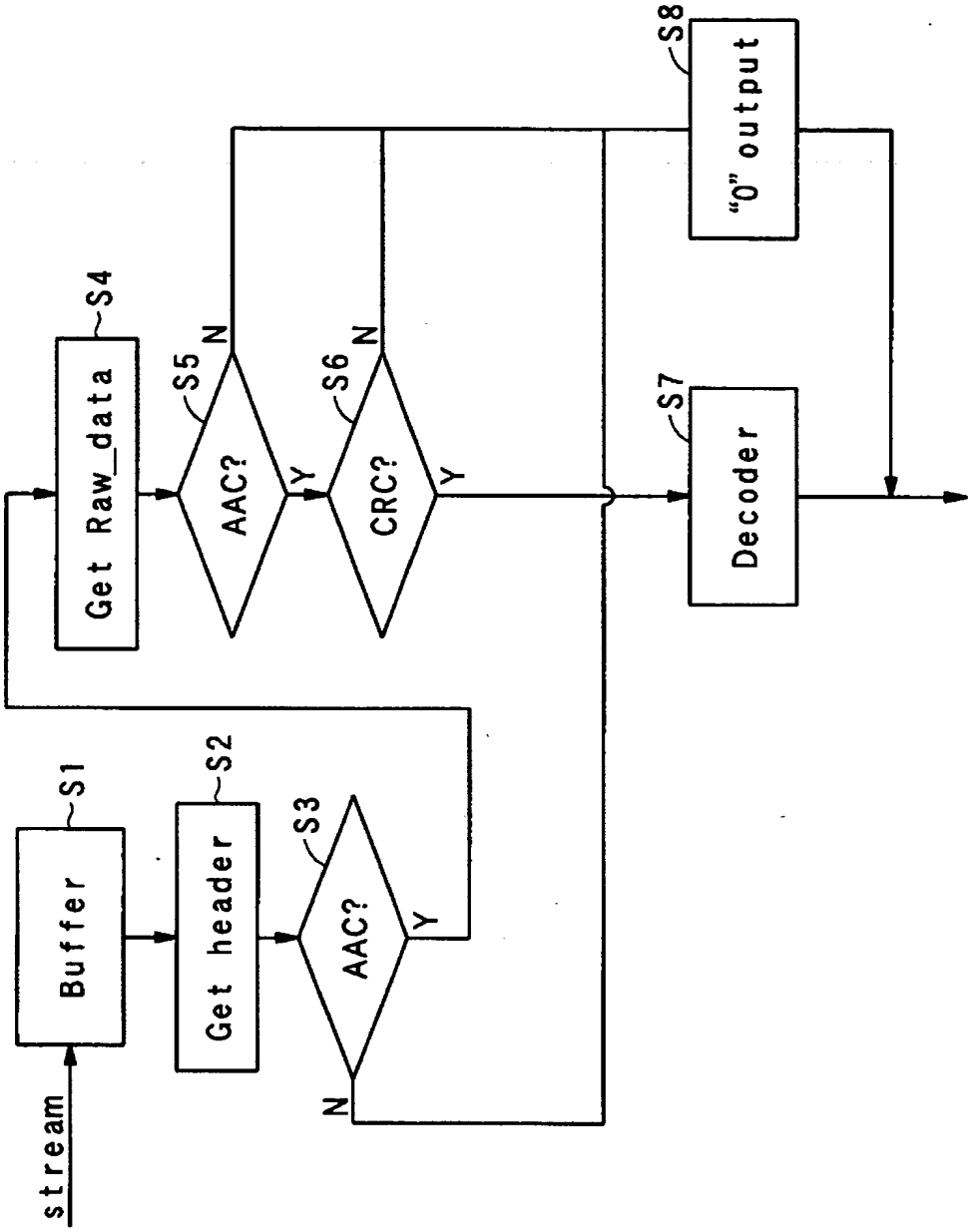


出証特２００３－３０５０５３２

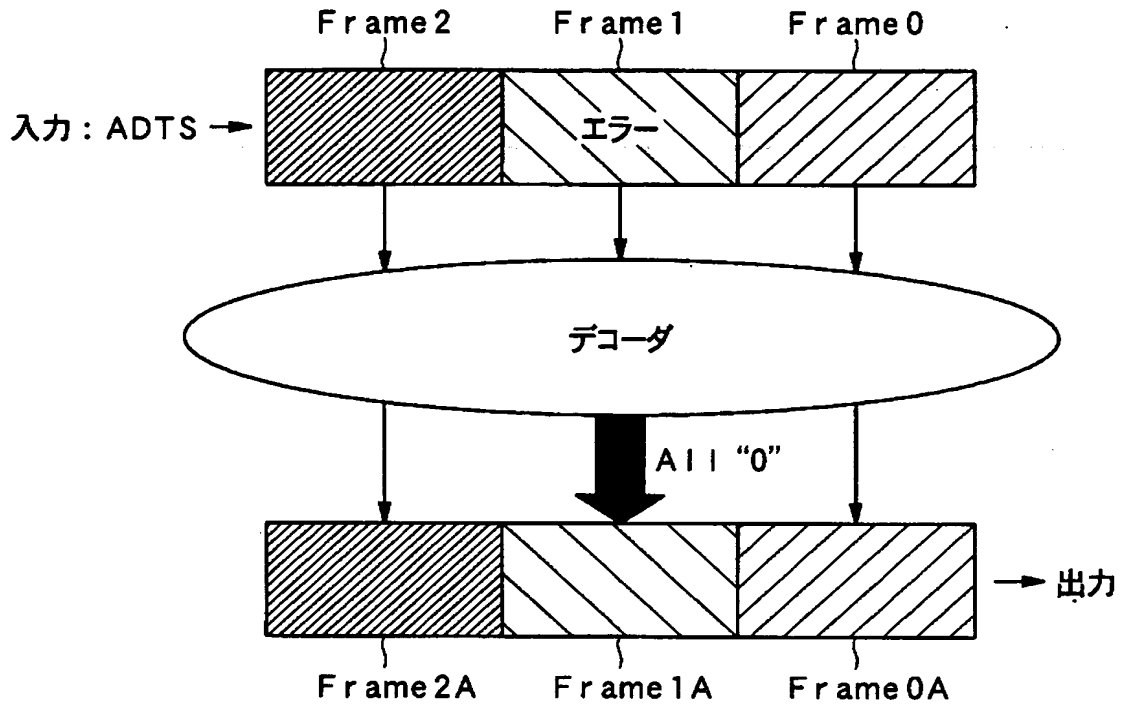
【図３】

Syntactic Element	ID name	Encoding	Abbreviation
single_channel_element	ID_SCE	0x0	SCE
channel_pair_element	ID_CPE	0x1	CPE
coupling_channel_element	ID_CCE	0x2	CCE
lfe_channel_element	ID_LFE	0x3	LFE
Data_stream_element	ID_DSE	0x4	DSE
program_config_element	ID_PCE	0x5	PCE
fill_element	ID_FIL	0x6	FIL
Terminator	ID_END	0x7	TERM

【図４】

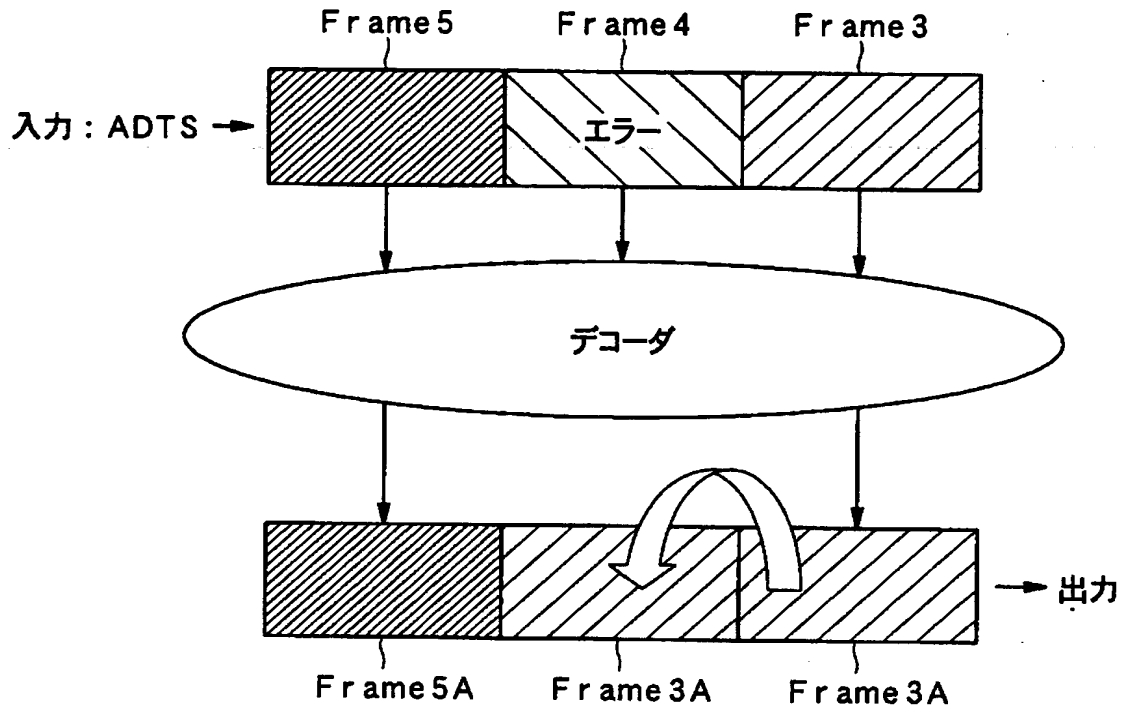

S8
"0" output
S4
S5
S6
S7
N
N
Get Raw_data
AAC?
CRC?
Decoder
Y
Y
S1
S2
S3
Buffer
Get header
AAC?
Y
N
stream

【図５】

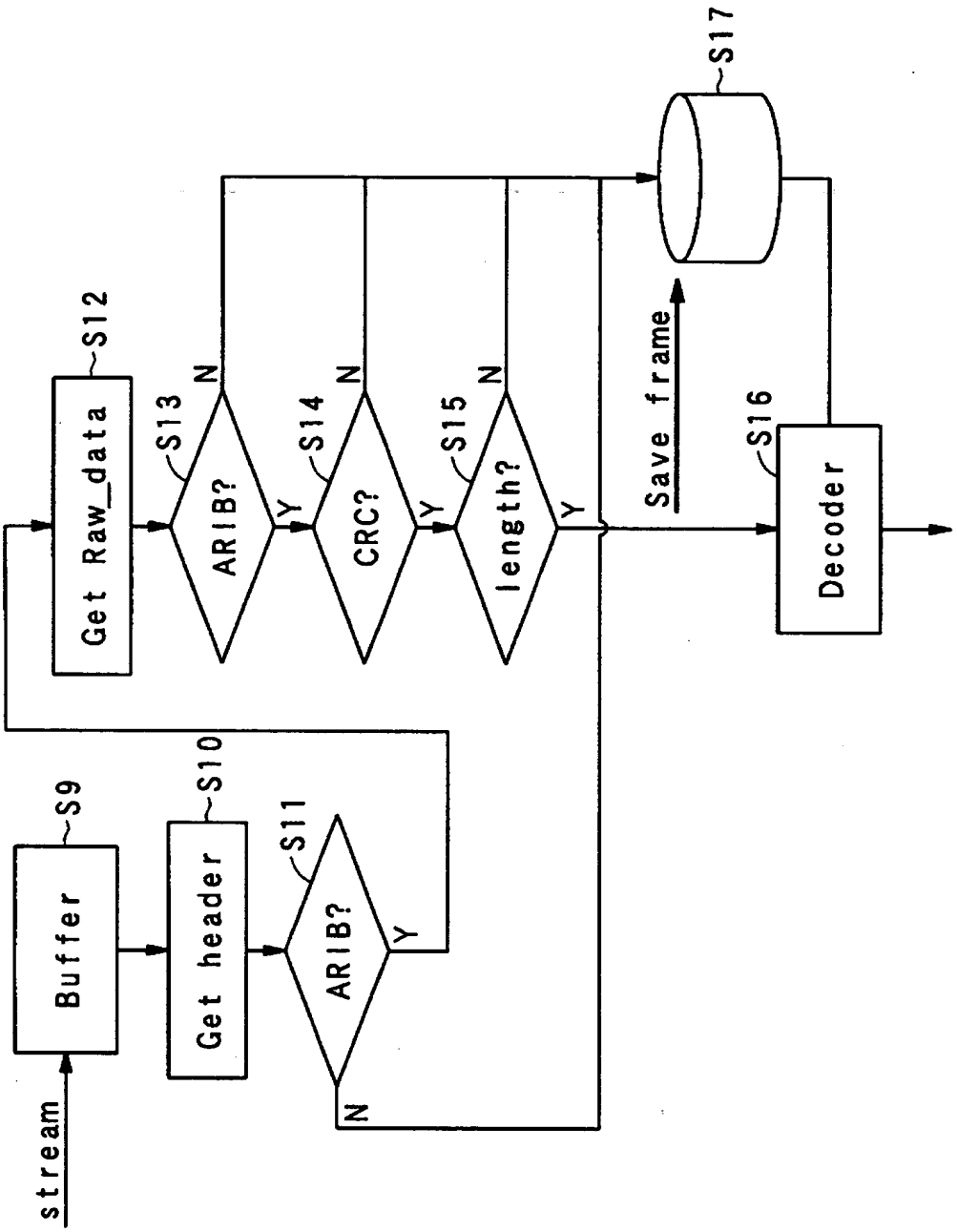


出証特２００３－３０５０５３２

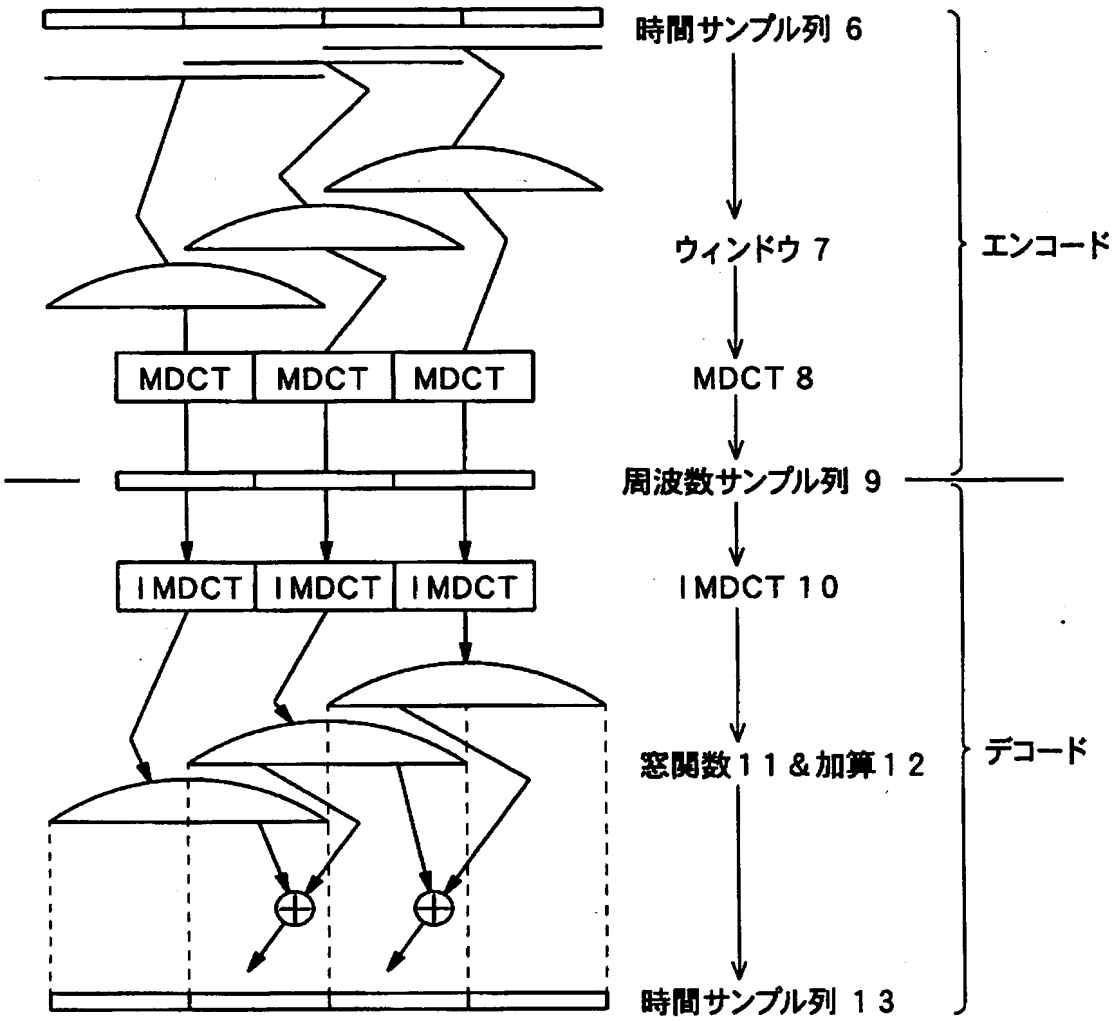
【図６】


Ｆｒａｍｅ５
Ｆｒａｍｅ４
Ｆｒａｍｅ３
入力：ＡＤＴＳ
エラー
デコーダ
出力
Ｆｒａｍｅ５Ａ
Ｆｒａｍｅ３Ａ
Ｆｒａｍｅ３Ａ

【図７】

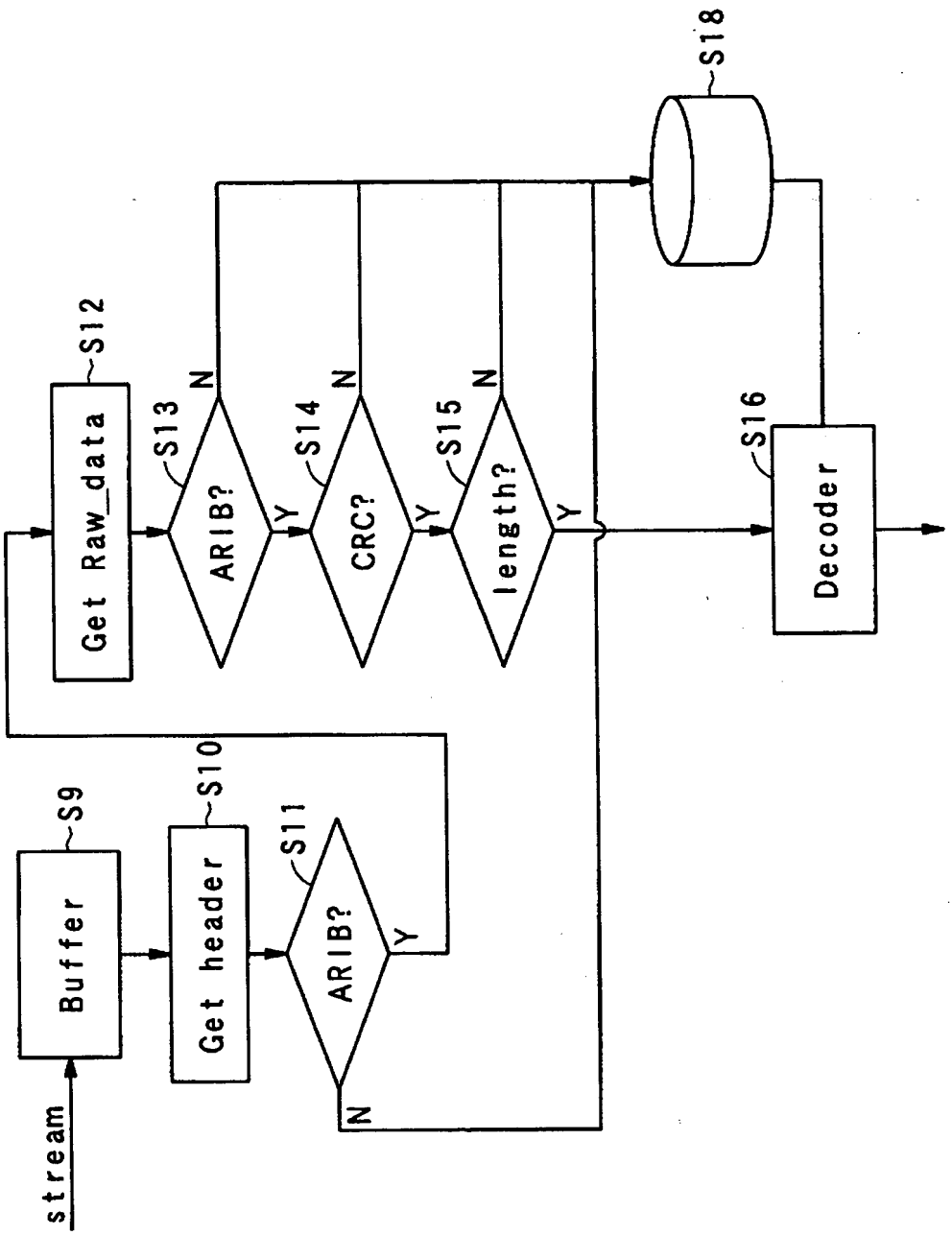

S17
S12
S13
S14
S15
S16
S9
S10
S11
Get Raw_data
ARIB?
CRC?
length?
Save frame
Decoder
Buffer
Get header
ARIB?
stream
N
Y

【図８】

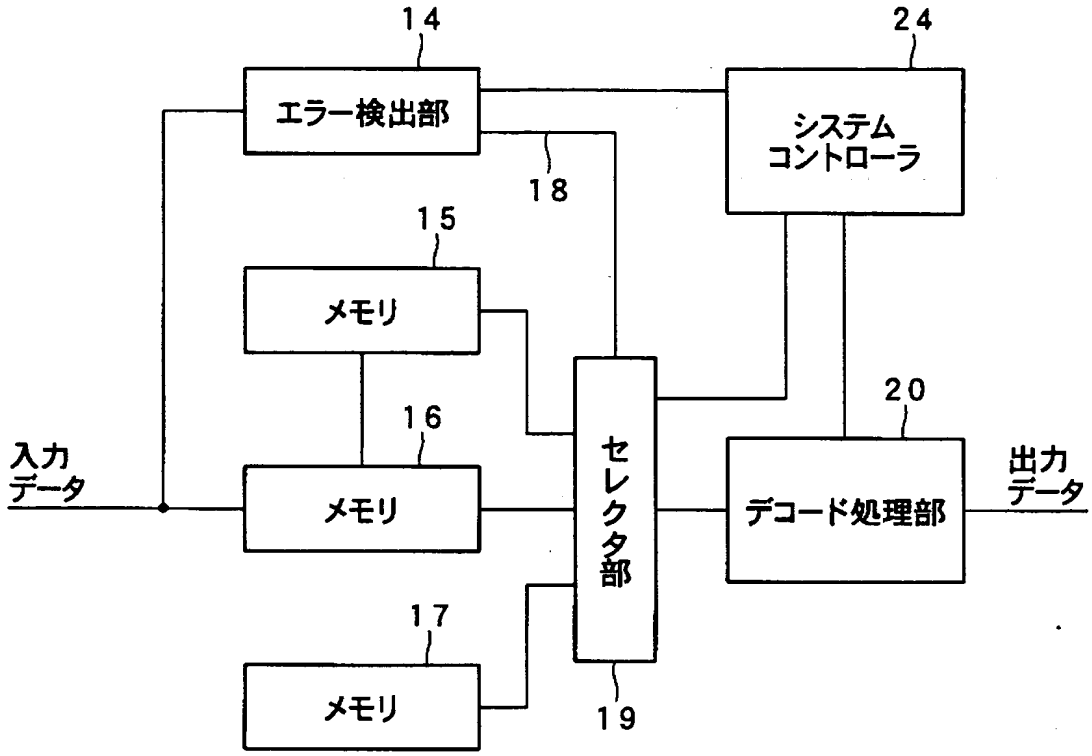


出証特２００３－３０５０５３２

【図９】


S18
S12
S13
S14
S15
S16
S9
S10
S11
Get Raw_data
ARIB?
CRC?
length?
Decoder
Buffer
Get header
ARIB?
stream
N
Y

【図１０】


14
エラー検出部
24
システム
コントローラ
18
15
メモリ
16
入力
データ
メモリ
セレクタ部
20
デコード処理部
出力
データ
17
メモリ
19

【書類名】　　要約書

【要約】

【課題】　ＣＲＣによるエラー検出よりもさらにエラー検出能力を高め、かつ、エラー検出時における出力音をより聞きやすい音に改善することを課題の一例としている。

【解決手段】　上記の課題を解決するために、請求項１に記載の発明は、符号化された音声データにエラーが発生したかどうかを検出するエラー検出手段と、前記符号化された音声データをデコードする際に窓関数を使用し、その結果を加算するデコード手段を備え、前記デコード手段は、前記エラー検出時にはエラーの発生していない直前の前記符号化された音声データをデコードすることを特徴とする。

【選択図】　図９

出　願　人　履　歴　情　報

識別番号　　〔０００００５０１６〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都目黒区目黒１丁目４番１号
氏　名　パイオニア株式会社